



# ECE-374-B: Lecture 0 - Logistics and Strings/Languages

---

**Instructor:** Abhishek Kumar Umrawal

January 16, 2024

University of Illinois at Urbana-Champaign

# Course Administration

---

# Instructional Staff

- **Instructor:**
  - Abhishek Kumar Umrawal
- **Teaching Assistants:**
  - Anthea (Anxue) Chen
  - Jack (Ziheng) Chen
  - Sung Woo Jeon
  - Hongyu Shen
  - Sumedh Vemuganti
  - Weiyang Wang
  - Haoyuan You
  - Hongbo Zheng
- **Office hours:** TBD, See course webpage
- **Contacting us:** Use private notes on Piazza to reach course staff. Direct email only for sensitive or confidential information.

## About your instructor – Basic info

- **Name:** Abhishek Kumar Umrawal
- **Webpage:** [ece.illinois.edu/about/directory/faculty/aumrawal](http://ece.illinois.edu/about/directory/faculty/aumrawal)
- **Email:** [aumrawal@illinois.edu](mailto:aumrawal@illinois.edu)
- **Office:** ECEB 3054
- **Office hours:** TBD

## About your instructor – Education

- **Purdue University**, Ph.D. in Industrial Engineering  
Dissertation: Machine Learning Algorithms for Influence Maximization on Social Networks
- **Purdue University**, MS in Economics
- **Indian Institute of Technology (IIT) Kanpur**, MS in Statistics

## About your instructor – Prior teaching experience

- **University of Maryland**, Visiting Lecturer of Computer Science and Electrical Engineering

## About your instructor – Research interests

### Core areas:

1. Combinatorial optimization
2. Approximation algorithms
3. Statistical learning theory
4. Reinforcement learning (RL)
5. Causal inference



# About your instructor – Research interests

## Core areas:

1. Combinatorial optimization
2. Approximation algorithms
3. Statistical learning theory
4. Reinforcement learning (RL)
5. Causal inference

## Applications:

1. Social networks
2. Promotional marketing
3. Intelligent transportation
4. Product recommendation

# About your instructor – Research interests

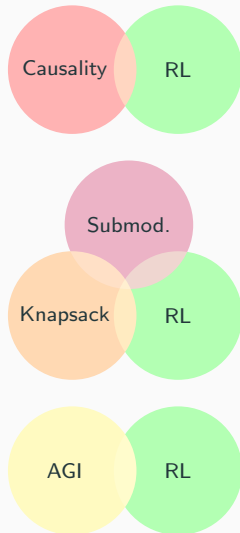
## Core areas:

1. Combinatorial optimization
2. Approximation algorithms
3. Statistical learning theory
4. Reinforcement learning (RL)
5. Causal inference

## Applications:

1. Social networks
2. Promotional marketing
3. Intelligent transportation
4. Product recommendation

## Intersections:



## Working with me on research

If you are interested in working with me then **please send me an email** with subject line 'Expressing interest in working with you on research' **with a brief description of your interests and skills** with no attachments. Please do so only after the second midterm so that you have spent enough time learning algorithms.

## Working with me on research

If you are interested in working with me then **please send me an email** with subject line 'Expressing interest in working with you on research' **with a brief description of your interests and skills** with no attachments. Please do so only after the second midterm so that you have spent enough time learning algorithms.

**Preferred** (but not required) **skills**:

- Mathematical thinking
- Probability and statistics
- Python programming – graphs, object-oriented programming, recursion, etc.
- Algorithms (you're doing it this semester!)

You may fill out [this](#) form to provide further information.

## Section A vs B

This semester, the two sections will be run completely **independently**.

- Different lectures.
- Different homeworks, quizzes, exams.
- Different grading policies.

## Section A vs B

This semester, the two sections will be run completely **independently**.

- Different lectures.
- Different homeworks, quizzes, exams.
- Different grading policies.

Section B will be in-person only. Recordings will be attempted but not guaranteed.

## Online resources

- **Webpage:** General information, announcements, homeworks, quizzes, course policies will be available at <https://ecealgo.com>.
- **Submission(Gradescope):** Written homework submission and grading, regrade requests. Exams will be uploaded there as well.
- **Communication(Piazza):** Announcements, online questions and discussion, contacting course staff (via private notes).
- **Gradebook (Canvas):** Announcements, online questions and discussion, contacting course staff (via private notes).

See course webpage for links.

**Important:** Check Piazza/course web page at least once each day.

## Discussion Sessions/Labs

- 50min problem solving session led by TAs.
- Two times a week.
- Go to your assigned discussion section.
- Bring pen and paper!



## Discussion Sessions/Labs

- 50min problem solving session led by TAs.
- Two times a week.
- Go to your assigned discussion section.
- Bring pen and paper!

Discussion sections will have questions that appear on the homework. If, you skip, you're just making more work for yourself later.

## Any questions

Again all policy information should be on course website:

<https://ecealgo.com>.

**Any questions?**

## **Over-arching course questions**

---

# High-Level Questions

This course introduces three distinct fields of computer science research:



- Computational complexity.
  - Given infinite time and a certain machine, is it possible to solve a given problem.
- Algorithms.
  - Given a deterministic Turing machine, how fast can we solve certain problems.
- Limits of computation.
  - Are there tasks that our computers cannot do and how do we identify these problems?

## Why not just focus on Algorithms?

When someone asks you, “How fast can you compute problem  $X$ ”, they are actually asking:

- Is  $X$  solvable using the deterministic Turing machines we have at our disposal?
- If it is solvable, can we find the solution efficiently (in poly-time)?
- If it is solvable but we don't have a poly time solution, what problem(s) is it most similar to?

# Course Structure

Course divided into three parts:

- Basic automata theory: finite state machines, regular languages, hint of context free languages/grammars, Turing Machines.
- Algorithms and algorithm design techniques.
- Undecidability and NP-Completeness, reductions to prove intractability of problems.

Week	Thematic Lecture	Workshop	Thematic Lecture	Self-Learn
Aug 26	Introduction and course goals Introduction and Theory 1. Wilson J. Search/Algs Lec 1+1 10 mins, written, absent	Mathematical Induction 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Language and regular expressions 1. Search/Algs Lec 1+1 10 mins, written	Regular expressions 1. Wilson J. Searching 1. Searching 1.
Aug 30 - Sep 2	Finite Automata, Regular Languages Wilson J. Introduction to Automata Theory Search/Algs Lec 1+1 10 mins, written	NP-Completeness 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Non-determinism, NFA's 1. Search/Algs Lec 1+1 10 mins, written	Language Complexity 1. Wilson J. Searching 1.
Sep 6	Existence of DFA, NFA, and regular expressions 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Search/Algs Lec 1+1 Wilson J. Searching 1. 10 mins, written	Pushdown Automata 1. Wilson J. Searching 1. 10 mins, written	Pushdown Automata 1. Wilson J. Searching 1.
Sep 13	Context free languages and automata 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Context free languages 1. Wilson J. Searching 1. 10 mins, written	Pushdown Automata 1. Wilson J. Searching 1. 10 mins, written	Pushdown Automata 1. Wilson J. Searching 1.
Sep 20	Introduction to Turing Machines 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Mathematical Induction 1. Wilson J. Searching 1. 10 mins, written	Mathematical Induction 1. Wilson J. Searching 1. 10 mins, written	Mathematical Induction 1. Wilson J. Searching 1.
Sep 27	Introduction to Algorithms 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Mathematical Induction 1. Wilson J. Searching 1. 10 mins, written	Divide and conquer: Selection, Recursion 1. Wilson J. Searching 1. 10 mins, written	Divide and Conquer 1. Wilson J. Searching 1.
Oct 4	Mathematical Induction 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Mathematical Induction 1. Wilson J. Searching 1. 10 mins, written	Divide and conquer: Selection, Recursion 1. Wilson J. Searching 1. 10 mins, written	Divide and Conquer 1. Wilson J. Searching 1.
Oct 11	More Dynamic Programming 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	More Dynamic Programming 1. Wilson J. Searching 1. 10 mins, written	Dynamic Programming 1. Wilson J. Searching 1. 10 mins, written	Dynamic Programming 1. Wilson J. Searching 1.
Oct 18	Graphs: Graphs, DFS, BFS and Shortest Path 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Graphs: Graphs, DFS, BFS and Shortest Path 1. Wilson J. Searching 1. 10 mins, written	Graphs: Graphs, DFS and BFS 1. Wilson J. Searching 1. 10 mins, written	Graphs: Graphs, DFS and BFS 1. Wilson J. Searching 1.
Oct 25	Mathematical Induction 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Mathematical Induction 1. Wilson J. Searching 1. 10 mins, written	Mathematical Induction 1. Wilson J. Searching 1. 10 mins, written	Mathematical Induction 1. Wilson J. Searching 1.
Nov 1	Algorithms and Complexity 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Algorithms and Complexity 1. Wilson J. Searching 1. 10 mins, written	Algorithms and Complexity 1. Wilson J. Searching 1. 10 mins, written	Algorithms and Complexity 1. Wilson J. Searching 1.
Nov 8	NP-Completeness 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	NP-Completeness 1. Wilson J. Searching 1. 10 mins, written	NP-Completeness 1. Wilson J. Searching 1. 10 mins, written	NP-Completeness 1. Wilson J. Searching 1.
Nov 15	Undecidability 1. Wilson J. Search/Algs Lec 1+1 10 mins, written	Undecidability 1. Wilson J. Searching 1. 10 mins, written	Undecidability 1. Wilson J. Searching 1. 10 mins, written	Undecidability 1. Wilson J. Searching 1.
Nov 22 - Dec 2	Final Exam Review for Midterm 1 Wilson J. Search/Algs Lec 1+1 10 mins, written	Final Exam Review for Midterm 1 Wilson J. Searching 1. 10 mins, written	Final Exam Review for Midterm 1 Wilson J. Searching 1. 10 mins, written	Final Exam Review for Midterm 1 Wilson J. Searching 1.
Dec 9	Midterm Review for Midterm 1 Wilson J. Search/Algs Lec 1+1 10 mins, written	Midterm Review for Midterm 1 Wilson J. Searching 1. 10 mins, written	Midterm Review for Midterm 1 Wilson J. Searching 1. 10 mins, written	Midterm Review for Midterm 1 Wilson J. Searching 1.

# Goals

RIY

- Algorithmic thinking.
- Learn/remember some basic tricks, algorithms, problems, ideas.
- Understand/appreciate limits of computation (intractability).
- Appreciate the importance of algorithms in computer science and beyond (engineering, mathematics, natural sciences, social sciences, ...).

# Formal languages and complexity

## (The Blue Weeks!)

---



# Why Languages?

First 5 weeks devoted to language theory.

# Why Languages?

First 5 weeks devoted to language theory.

But why study languages?

# Multiplying Numbers

Consider the following problem:

**Problem** Given two  $n$ -digit numbers  $x$  and  $y$ , compute their product.

## **Grade School Multiplication**

Compute "partial product" by multiplying each digit of  $y$  with  $x$  and adding the partial products.

$$\begin{array}{r} 3141 \\ \times 2718 \\ \hline 25128 \\ 3141 \\ 21987 \\ 6282 \\ \hline 8537238 \end{array}$$



## Fast Multiplication

- $O(n^{1.58})$  time [Karatsuba 1960] disproving Kolmogorov's belief that  $\Omega(n^2)$  is best possible.
- $O(n \log n \log \log n)$  [Schönhage-Strassen. 1971].  
**Conjecture:**  $O(n \log n)$  time possible.
- $O(n \log n \cdot 2^{O(\log^* n)})$  time [Furer 2008].
- $O(n \log n)$  [Harvey-van der Hoeven 2019].

$$n \leq \underline{n \log n} \leq \underline{n^2}$$

Can we achieve  $O(n)$ ? No lower bound beyond trivial one!

## Equivalent Complexity

→  $O(n \log n)$

Does this mean multiplication is as complex as another problem that has a  $O(n \log n)$  algorithm like sorting/QuickSort?

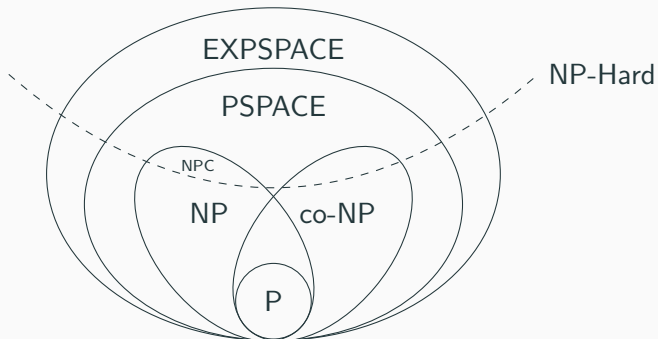
## Equivalent Complexity

Does this mean multiplication is as complex as another problem that has a  $O(n \log n)$  algorithm like sorting/QuickSort?

How do we compare? The two problems have:

- Different inputs (two numbers vs n-element array).
- Different outputs (a number vs n-element array).
- Different entropy characteristics (from a information theory perspective).

An algorithm has a runtime complexity.

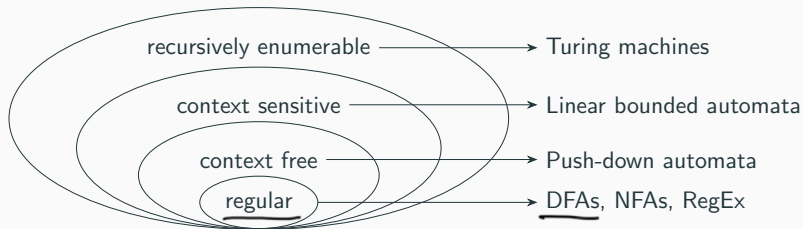




# Languages, Problems and Algorithms ... oh my! III

A problem has a complexity class!

Recognized by:



Problems do not have run-time since a problem  $\neq$  the algorithm used to solve it. *Complexity classes are defined differently.*

How do we compare problems? What if we just want to know if a problem is “computable”.

## Definition

1. An algorithm is a step-by-step way to solve a problem.
2. A problem is some question that we'd like answered given some input. It should be a decision problem of the form "Does a given input fulfill property X."
3. A Language is a set of strings. Given a alphabet,  $\Sigma$  a language is a subset of  $\Sigma^*$ .

## Definition

1. An **algorithm** is a step-by-step way to solve a problem.
2. A **problem** is some question that we'd like answered given some input. It should be a decision problem of the form "Does a given input fulfill property X."
3. A **Language** is a set of strings. Given a alphabet,  $\Sigma$  a language is a subset of  $\Sigma^*$ . A language is a formal realization of this problem. For problem X, the corresponding language is:

$L = \{w \mid w \text{ is the encoding of an input } y \text{ to problem } X \text{ and the answer to input } y \text{ for a problem } X \text{ is "YES" } \}$

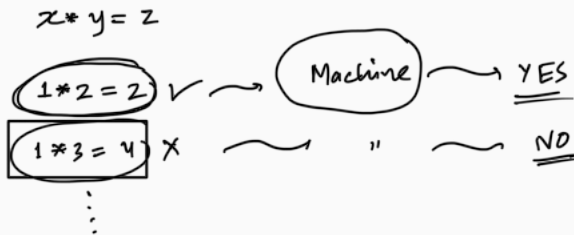
A decision problem X is "YES" is the string is in the language.

# Language of multiplication

How do we define the multiplication problem as a language?

Define L as language where inputs are separated by comma and output is separated by —.

Machine accepts a  $x*y=z$  if  $'x*y=z'$  is in L. Rejects otherwise.



$$\Sigma = \{0, 1\}$$

$$w = 0101$$

$$L = \{01, 001\}$$

## Language of multiplication

How do we define the multiplication problem as a language?

Define L as language where inputs are separated by comma and output is separated by —.

Machine accepts a  $x*y=z$  if " $x*y—z$ " is in L. Rejects otherwise.

$$\underline{L_{MULT2}} = \left\{ \begin{array}{ccc} \underline{1 \times 1|1}, & \underline{1 \times 2|2}, & \underline{1 \times 3|3}, \dots \\ \underline{2 \times 1|2}, & \underline{2 \times 2|4}, & \underline{2 \times 3|6}, \dots \\ \vdots & \vdots & \vdots \\ \underline{n \times 1|n}, & \underline{n \times 2|2n}, & \underline{n \times 3|3n}, \dots \end{array} \right\} \quad (1)$$

$$1 \times 2|9 \notin L_{MULT2}$$

## Language of sorting

We do the same thing for sorting.

Define L as language where inputs are separated by comma and output is separated by —.

Machine accepts a  $[i_1, i_2, \dots]$  =  $sort(\{i_1, i_2, \dots\})$  if " $x[]—z[]$ " is in L. Rejects otherwise.

## Language of sorting

We do the same thing for sorting.

Define  $L$  as language where inputs are separated by comma and output is separated by  $|$ .

Machine accepts a  $[i_1, i_2, \dots] = \text{sort}(\{i_1, i_2, \dots\})$  if " $x|z$ " is in  $L$ . Rejects otherwise.

$$\underline{L_{\text{Sort2}}} = \left\{ \begin{array}{ccc} \underline{1, 1} | \underline{1, 1} & 1, 2 | 1, 2 & 1, 3 | 1, 3, \dots \\ \underline{2, 1} | \underline{1, 2} & 2, 2 | 2, 2, & 2, 3 | 2, 3, \dots \\ \vdots & \vdots & \vdots \\ \textcircled{n} 1 | 1, n, & n, 2 | 2, n, & n, 3 | 3, n, \dots \end{array} \right\} \quad (2)$$

$$2, 1 | 2, 1 \notin L_{\text{Sort2}}$$

## Language of sorting

We do the same thing for sorting.

Define  $L$  as language where inputs are separated by comma and output is separated by  $|$ .

Machine accepts a  $[i_1, i_2, \dots] = \text{sort}(\{i_1, i_2, \dots\})$  if " $x|z$ " is in  $L$ . Rejects otherwise.

$$L_{\text{Sort2}} = \left\{ \begin{array}{lll} 1, 1|1, 1 & 1, 2|1, 2 & 1, 3|1, 3, \dots \\ 2, 1|1, 2, & 2, 2|2, 2, & 2, 3|2, 3, \dots \\ \vdots & \vdots & \vdots \\ n, 1|1, n, & n, 2|2, n, & n, 3|3, n, \dots \end{array} \right\} \quad (2)$$

If the same type of machine can recognize both languages, then that gives us an upperbound to their hardness.



**How do we formulate languages?**

---

# Strings

---

# Alphabet

An alphabet is a finite set of symbols.

Examples of alphabets:

- $\Sigma = \{0, 1\}$ ,
- $\Sigma = \{a, b, c, \dots, z\}$ ,
- ASCII.
- UTF8.
- $\Sigma =$   
 $\{\langle(w)forward\rangle, \langle(a)strafe\ left\rangle, \langle(s)back\rangle, \langle(d)strafe\ right\rangle\}$

# String Definition

$$|\epsilon| = 0 \quad |\{\epsilon\}| = 1 \quad \epsilon : \text{empty string}$$

## Definition

1. A string/word over  $\Sigma$  is a (finite) sequence of symbols over  $\Sigma$ .  
For example, '0101001', 'string', '⟨moveback⟩⟨rotate90⟩'
2.  $x \cdot y \equiv xy$  is the concatenation of two strings
3. The length of a string  $w$  (denoted by  $|w|$ ) is the number of symbols in  $w$ . For example,  $|101| = 3$ ,  $|\epsilon| = 0$
4. For integer  $n \geq 0$ ,  $(\Sigma^n)$  is set of all strings over  $\Sigma$  of length  $n$ .  
 $\Sigma^*$  is the set of all strings over  $\Sigma$ .
5.  $(\Sigma^*)$  set of all strings of all lengths including empty string.

**Question:** What is  $\{ '0', '1' \}^*$ ?

$$\{0, 1\}^* = \{ \epsilon, 0, 1, 00, 010, \dots \}$$

# Emptiness

- $\epsilon$  is a string containing no symbols. It is not a set
- $\{\epsilon\}$  is a set containing one string: the empty string. It is a set, not a string.
- $\emptyset$  is the empty set. It contains no strings.

**Question:** What is  $\{\emptyset\}$ ?

$$|\{\emptyset\}| = 1$$

# Concatenation and properties



- If  $x$  and  $y$  are strings then  $xy$  denotes their concatenation.
- **Concatenation** defined recursively :
  - $xy \equiv y$  if  $x \equiv \epsilon$
  - $xy = a(wy)$  if  $x = aw$
- $xy$  sometimes written as  $x \cdot y$ .
- concatenation is **associative**:  $(uv)w = u(vw)$  hence write  $uvw \equiv (uv)w = u(vw)$
- **not commutative**:  $uv$  not necessarily equal to  $vu$
- The identity element is the empty string  $\epsilon$ :

$$\underline{\epsilon u = u \epsilon = u.}$$

# Substrings, prefixes, Suffixes

1010101  
└──┬──┘  
└──┘  
101

## Definition

$v$  is **substring** of  $w$   $\iff$  there exist strings  $x, y$  such that  
 $w = xvy$ .

- If  $x = \epsilon$  then  $v$  is a **prefix** of  $w$
- If  $y = \epsilon$  then  $v$  is a **suffix** of  $w$

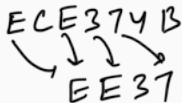
$w = 10101$   
└──┬──┘  
└──┘  
101 is prefix of  $w$   
01 is a suffix of  $w$

# Subsequence

A subsequence of a string  $w[1\dots n]$  is either a subsequence of  $w[2\dots n]$  or  $w[1]$  followed by a subsequence of  $w[2\dots n]$ .

## Example

EE37 is a subsequence of ECE374B



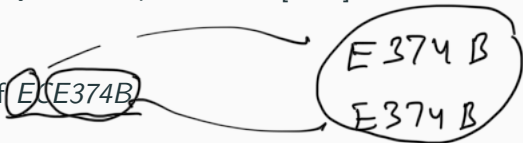


# Subsequence

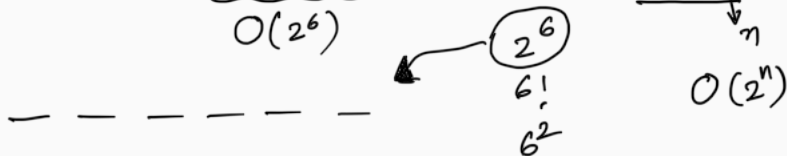
A subsequence of a string  $w[1\dots n]$  is either a subsequence of  $w[2\dots n]$  or  $w[1]$  followed by a subsequence of  $w[2\dots n]$ .

## Example

$EE37$  is a subsequence of  $EE374B$



**Question:** How many sub-sequences are there in a string  $|w| = 6$ ?



## String exponent

### Definition

If  $w$  is a string then  $w^n$  is defined inductively as follows:

$$\underline{w^n = \epsilon \text{ if } n = 0}$$

$$\underline{w^n = ww^{n-1} \text{ if } n > 0}$$


**Question:**  $(ha)^3 = .hahaha$

$$(ha)^0 = \epsilon$$

## Rapid-fire questions -strings

Answer the following questions taking  $\Sigma = \underline{\{0, 1\}}$ .

1. What is  $\Sigma^0$ ?  $\{\epsilon\}$

2. How many elements are there in  $\Sigma^n$ ?   $\Sigma^1 = \{0, 1\}$   
 $\Sigma^2 = \{00, 01, 11\}$

3. If  $|u| = 2$  and  $|v| = 3$  then what is  $|u \cdot v|$ ?  $2+3=5$   $\therefore$   $10?$   
(Find the pattern)

4. Let  $u$  be an arbitrary string in  $\Sigma^*$ . What is  $\epsilon u$ ? What is  $u \epsilon$ ?

$$\begin{pmatrix} \epsilon u = u \\ u \epsilon = u \end{pmatrix}$$

# Languages

---

## Definition

A **language**  $L$  is a set of strings over  $\Sigma$ . In other words  $L \subseteq \Sigma^*$ .

## Definition

A **language**  $L$  is a set of strings over  $\Sigma$ . In other words  $L \subseteq \Sigma^*$ .

Standard set operations apply to languages.

- For languages  $A, B$  the **concatenation** of  $A, B$  is  $AB = \{xy \mid x \in A, y \in B\}$ .
- For languages  $A, B$ , their **union** is  $A \cup B$ , **intersection** is  $A \cap B$ , and **difference** is  $A \setminus B$  (also written as  $A - B$ ).
- For language  $A \subseteq \Sigma^*$  the **complement** of  $A$  is  $\bar{A} = \Sigma^* \setminus A$ .

# Set Concatenation

## Definition

Given two sets  $X$  and  $Y$  of strings (over some common alphabet  $\Sigma$ ) the concatenation of  $X$  and  $Y$  is

$$\underline{XY = \{xy \mid x \in X, y \in Y\}} \quad (3)$$

**Question:**  $X = \{ECE, CS, \}$ ,  $Y = \{340, 374\} \implies$   
 $XY = .$

DIY

**Definition**

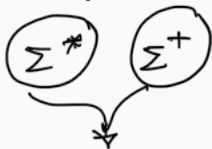
1.  $\Sigma^n$  is the set of all strings of length  $n$ . Defined inductively:

$$\Sigma^n = \{\epsilon\} \text{ if } n = 0$$

$$\Sigma^n = \Sigma \Sigma^{n-1} \text{ if } n > 0$$

2.  $\Sigma^*$  =  $\bigcup_{n \geq 0} \Sigma^n$  is the set of all finite length strings

3.  $\Sigma^+$  =  $\bigcup_{n \geq 1} \Sigma^n$  is the set of non-empty strings.

**Definition**

A **language**  $L$  is a set of strings over  $\Sigma$ . In other words  $L \subseteq \Sigma^*$ .

**Question:** Does  $\Sigma^*$  have strings of infinite length?

NO!

$$\Sigma^n = \underbrace{\Sigma}^1 \underbrace{\Sigma^{n-1}}$$

$$\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$$



## Rapid-Fire questions - Languages

### Problem

Consider languages over  $\Sigma = \{0, 1\}$ .

1. What is  $\emptyset^0$ ?
2. If  $|L| = 2$ , then what is  $|L^4|$ ?
3. What is  $\emptyset^*$ ,  $\{\epsilon\}^*$ ?
4. For what  $L$  is  $L^*$  finite?
5. What is  $\emptyset^+$ ?
6. What is  $\{\epsilon\}^+$ ?



# Terminology Review

Let's review what we learned.

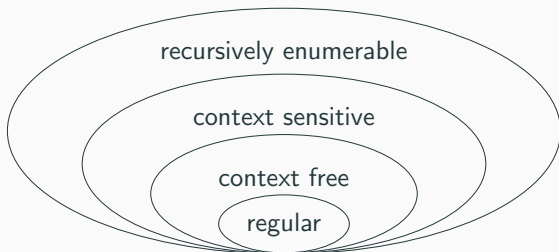
- A character( $a, b, c, x$ ) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A alphabet( $\Sigma$ ) is a set of characters
- A string( $w$ ) is a sequence of characters
- A language( $A, B, C, L$ ) is a set of strings

# Terminology Review

Let's review what we learned.

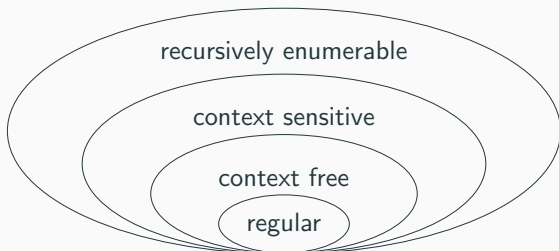
- A **character**( $a, b, c, x$ ) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A **alphabet**( $\Sigma$ ) is a set of characters
- A **string**( $w$ ) is a sequence of characters
- A **language**( $A, B, C, L$ ) is a set of strings
- A **grammar**( $G$ ) is a set of rules that defines the strings that belong to a language

## Languages: easiest, easy, hard, really hard, really<sup>n</sup> hard



- Regular languages.
  - Regular expressions.
  - DFA: Deterministic finite automata.
  - NFA: Non-deterministic finite automata.
  - Languages that are not regular.
- Context free languages (stack).
- Turing machines: Decidable languages.
- TM Undecidable/unrecognizable languages (halting theorem). 38

# Languages: easiest, easy, hard, really hard, really<sup>n</sup> hard



- Regular languages.
  - Regular expressions. ← **Next lecture**
  - DFA: Deterministic finite automata.
  - NFA: Non-deterministic finite automata.
  - Languages that are not regular.
- Context free languages (stack).
- Turing machines: Decidable languages.
- TM Undecidable/unrecognizable languages (halting theorem). 38

## That's it for now

Check the course website (<https://ecealgo.com>) for lab and hw schedule.